APPLICATION FOR UNITED STATES LETTERS PATENT

by

MAURICIO LOPEZ

AND

MARK KIRKPATRICK

for a

SYSTEM AND METHOD FOR ANALYZING COMPUTER APPLICATIONS IN REAL-TIME

SHAW PITTMAN LLP
1650 Tysons Boulevard
McLean, VA 22102-4859
(703) 770-7900
Attorney Docket No.: BS01-320

SYSTEM AND METHOD FOR ANALYZING EXECUTING COMPUTER APPLICATIONS IN REAL-TIME

BACKGROUND

Field of the Invention

[0001] The present invention relates generally to the field of diagnostic tools for computer software. More specifically, the present invention relates to real-time analysis of a computer application while it is executing without terminating or interrupting the application's execution.

Background of the Invention

[0002] Web-based software applications often execute on application server platforms. Examples of such application server platforms include systems such as iPlanet or Weblogic. Due to their nature, it is desirable to keep web-based applications highly available. Thus, minimizing their downtime is a critical goal. However, this goal can pose significant problems in the trouble-shooting and maintenance environment of most IT facilities. Downtime is often required to diagnose and correct software errors or to upgrade systems. Downtime to maintain systems or find and repair errors can have enormous impacts on IT facilities' operations and viability.

[0003] Conventional debugging tools do not adequately solve the problem of analyzing an application during its normal unimpeded execution. These tools are generally limited to assisting users in locating software errors in the logic of written computer software. When used, these tools do not allow analysis of the computer software operating in its normal mode of operation. Rather, the operation of the

software is controlled by the debugging tool. Thus, the debugging tool interferes with and/or interrupts the natural execution of the software. An immediate difficulty with using such tools is diagnosing performance problems. This is because performance problems generally arise from the effect of external system dynamics acting software during its normal operation, rather than from erroneous computer software. Because conventional debugging tools interfere with an application's natural operation, they do not provide insight into what problems are giving rise to performance-related issues. Thus, conventional debugging tools tend to be of little value for addressing performance-related problems. Consequently, there is a need for a computer software diagnostic tool that can provide users with information to analyze executing web-based applications without terminating or seriously impeding their normal operation.

SUMMARY OF THE INVENTION

[0004] The present invention solves the foregoing problems in the art by providing a user with an interface that allows the user to analyze web-based applications while they are executing. The analysis is performed without interrupting or terminating the application being analyzed. By providing insight into the operation of an executing application, the present invention allows the user to analyze and understand causes of performance issues and/or determine the source of errors that may be present in the applications.

[0005] Preferably, an object shell console is used to connect to an executing application. The object shell console extracts software components from the executing application and displays these to the user. For example, the object shell console can obtain and display information regarding the classes and methods

comprising a particular application, the value of the variables used in those methods, the execution time of methods, the number of times that an instance of a particular method or class is invoked, and other data that is useful in analyzing performance of an executing application.

[0006] In one embodiment, the present invention is a system for analyzing a computer application while it is executing without terminating or interrupting the application. An application to be analyzed executes on a computer such as an application server. An object shell console executing on an administration client attaches to an application to be analyzed that is in execution so that it can extract information from the application without interrupting the application or causing the application to terminate. The information is presented to a user in a graphical user interface presented by the object shell console. The user can also obtain more detailed information about the application that is being analyzed.

[0007] In another embodiment, the present invention is a method for analyzing a computer application while it is executing. The method includes the steps of connecting to the computer application and extracting information from the computer application without interrupting or terminating the computer application. The method also includes the step of displaying the information to a user in a graphical user interface. In addition, more detailed information can be provided to the user if such information exists.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] Figure 1 is a schematic diagram a system for analyzing computer applications according to an embodiment of the present invention.

[0009] Figure 2 is a flow chart illustrating a method for analyzing computer applications according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0010] The cause of many performance-related problems in web-based applications is often not immediately apparent. This is because, the problem is not with the software itself, but with a change to the system that occurs due to the dynamic nature of web-based systems. Consequently, the problem cannot be analyzed using conventional diagnostic tools because such tools do not allow the application being analyzed to run without interruption. Furthermore, it is likely that the performance-related problem may not be revealed when the application is executed in this manner. Without the present invention, this kind of error could be extremely difficult to detect.

[0011] Figure 1 is a schematic diagram of a system for analyzing computer applications according to an embodiment of the present invention. An object shell console 102 executes on an administration client 104. Administration client 104 can be any computer that can execute an object shell console 102 having the functionality described herein. Object shell console 102 attaches to an application executing on an application server 106 so that the application can be analyzed. Importantly, the connection does not interrupt or terminate the application to be analyzed.

[0012] Exemplary applications 110, 112, 114 and 116 are illustrated executing on an application server 106. One application, for example, is an order application 110. Order application 110 helps handle orders from customers. To support its function, order application 110 is coupled to an order database 118. Order database 118

contains order information required to support order application 110. For example, order database 118 contains inventory data, supplier data and customer data.

[0013] In an embodiment of the present invention, object shell console 102 uses attributes of an interpreter programming language, such as the Java programming language, to connect to an application that is to be analyzed without terminating the application. Once connected, object shell console 102 can provide information to a user to analyze the executing application without terminating or impeding the executing application. Object shell console 102 can provide this capability by connecting to the Java virtual machine (JVM) on which the application to be analyzed is executing.

[0014] One exemplary way for making this connection is to use Java remote method invocation (RMI). Java RMI is a well-known tool that can be used to access one JVM from another JVM. Using Java RMI allows remote invocation and execution of applications and methods. To invoke a remote application (or method), Java RMI creates a thread to the other application or method. Creation of the new thread using Java RMI occurs without interrupting or terminating the executing application. Thus, the application to which the connection is made continues to execute normally.

[0015] The application that is connected to and invoked can be either local or remote to administration client 104. Once the connection is established, object shell console 102 has access to the internal structure of the invoked application. This access is allowed by a feature of the Java programming known as introspection. Introspection was developed for Java Beans to allow for integrated development environments (IDEs) to visually manipulate graphical components to build applications. The

classes that are used for introspection are Class, Method and Field. These classes are also the components of an object. When object shell console 102 retains a reference to a running object by creating the thread, these classes are used to extract the fields and execute the methods of the object. The fields of the object can be manipulated to create different behaviors in the object. Methods can be re-executed with argument values supplied by an operator or system administrator. The values returned by the method's execution can be displayed for analysis.

[0016] Object shell console 102 gains this access without terminating or interrupting the invoked application. Thus, once connected, object shell console 102 can determine classes and methods that are present in the invoked application without terminating or interrupting the invoked application.

[0017] After the thread to the application is created, the application can be analyzed. One kind of data in the application that is accessible to object shell console 102 via the new thread is the set of methods that comprise the application. Using this information, object shell console 102 can present a list of methods that are running in the application to a user of the present invention. Preferably, the methods are listed hierarchically. Primary categories of methods and/or classes are listed. Subcategories of the primary categories allow for more detailed analysis. For example, one or more of the classes comprising the application can be listed. Under each class, methods comprising the class can be listed. Referring to Figure 1, for example, order application has a database class. Database class has read(), write() and init()methods.

[0018] In addition, object shell console 102 allows a user to examine the operation of an application or method. To assist the user, object shell console 102 can display any information about the executing application or method. The information that is displayed can be selected by the user, pre-configured or a combination of user-selected and pre-configured information. For example, object shell console 102 can display a list of variables by name and/or value available for a method in a class. In addition, object shell console 102 can display a list of arguments by name and/or value that are passed to or from a method or used by a class. Further, object shell console 102 can display the computer code of the application, or any method or class in the application.

[0019] Moreover, object shell console 102 can provide other information regarding the objects, classes or methods comprising an application. For example, object console 102 can display the number of times the method is invoked. An unexpectedly large number of invocations of a particular method indicates the presence of a possible loop condition. Further, object shell console 102 can display the execution time of a particular method. A large value of this time might indicate a particular performance problem that needs to be addressed. For example, an unusually long database write time might indicate problems with database 118, or that there are external influences that need to be checked that might be affecting its behavior.

[0020] After extracting the information, object shell console 102 formats the information so that it can be displayed in a graphical user interface (GUI). For

example, object shell console 102 presents a list of all the classes and methods that are available within order application 110 to the user.

[0021] The user can obtain more detailed information from the information displayed on the GUI. For example, the user can select a displayed method name to select a particular method, and request more detailed information corresponding to that method. Information, including the name and value of variables used in the method, the name and value of arguments passed to or from the method, the number of times the method is invoked and execution time of the method. Further, any methods comprising the selected method can be shown to the user.

[0022] An example will help to clarify the present invention. Assume there is an indication of a performance problem with order database 118. Such indications include errors in error logs corresponding to order database 118 and complaints by users of order database 118. Using object shell console 102, a user of the present invention attaches to order application 110. As described above, this attachment is preferably achieved through creation of a new thread for the application using Java RMI. The new thread gives object shell console 102 access to the internal structure of order application 10.

[0023] Once attached to order application 110, object shell console 102 extracts information from the newly created thread such as classes and methods comprising order application 110. As described above, this information is presented to the user in the GUI of object shell console102. In the present example, order application 110 includes a database class. The database class is comprised of 3 methods, read(),

write() and init(). The application name, class name and methods are displayed to the user in object shell console 112's GUI.

[0024] Figure 2 is a method for analyzing an executing computer application without termination or interruption of the executing application according to the present invention. The method can be executed by an administration client as described above. In step 201, a user selects an application for analysis. In step 202, the present invention attaches to an application to be analyzed. As described above, this connection can be made by creating a thread to the executing application. In one embodiment of the present invention, Java RMI is used to create the thread.

[0025] In step 204, information related to the executed application is extracted using the created thread. This information can include any information that a user can use to analyze the executing application. For example, the information can include methods and classes comprising the executing application. Further the information can include variable names, variable values, argument names, argument values, execution time, number of time a method is invoked and other information about the executing application.

[0026] In step 206, the information is presented to the user. Preferably, the information is formatted for presentation in a graphical user interface (GUI). In one embodiment of the present invention, the information is presented to the user in a hierarchical fashion. For example, classes comprising the application are listed. under each method, any methods in the class are listed. This hierarchy can continue for as many levels as are required to present the information to the user.

[0027] In step 208, addition detailed information can be extracted from the application if the user requests it. For example, the user may desire more information about a particular method. The user can double click the method name to obtain information about variable and/or argument names and/or values. The information is obtained in response to this request and displayed to the user in step 210. Preferably, the additional detailed information is displayed on the GUI.

[0028] The foregoing disclosure of the preferred embodiments of the present invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many variations and modifications of the embodiments described herein will be apparent to one of ordinary skill in the art in light of the above disclosure. The scope of the invention is to be defined only by the claims appended hereto, and by their equivalents.

[0029] Further, in describing representative embodiments of the present invention, the specification may have presented the method and/or process of the present invention as a particular sequence of steps. However, to the extent that the method or process does not rely on the particular order of steps set forth herein, the method or process should not be limited to the particular sequence of steps described. As one of ordinary skill in the art would appreciate, other sequences of steps may be possible. Therefore, the particular order of the steps set forth in the specification should not be construed as limitations on the claims. In addition, the claims directed to the method and/or process of the present invention should not be limited to the performance of their steps in the order written, and one skilled in the art can readily appreciate that

the sequences may be varied and still remain within the spirit and scope of the present invention.